Your application team is developing an application that uses different Docker images to test developers' code. So far, the images have been built on every server locally. The application team has asked you to deploy a Docker image registry on Kubernetes, which will be storing Docker images used by their application in future.

Your task

Write a Kubernetes object according to the expectations listed below:

* Docker images registry will be deployed as a single replica;
* Pod runs a single container with image registry :2.7.1;
* Container stores images under /var/iib/registry. Mount a Persistent Volume to store data;
* Additional configuration for the Persistent Volume:

o Set access modes tO ReadWriteOnce; o Persistent Volume should request 1 eeGi storage size;

* The container exposes port 5000;
* The container must have configured a LivenessProbe, which will attempt to open a socket to your container on port 5000 (use tcpsocket);
* Additional configurations on the LivenessProbe:

o Set initial probe delay to 10 seconds;

* The container must have configured a ReadinessProbe, which will attempt to open a socket to your container on port 5000 (use tcpsocket);
* Additional configurations on the ReadinessProbe:

o Set initial probe delay to 10 seconds.

Additionally, write a Service with type LoadBaiancer, which exposes the above deployment on port 5000.

For this test, assume that:

* You will use either statefuiset or Deployment object definition from apiGroup apps/vi to create the Deployment;
* If necessary, you may use Persistentvoiumeciaim from apiGroup vi
* You can use tcpsocket requests in probes;
* The Deployment will be created in the default namespace (it is not expected to define its own namespace);
* Your solution will be applied using kubectl apply -n default -f solution.yami;
* The file you are editing should be written as a valid YAML file;
* The Deployment will run on Kubernetes v1.16.

Write a Kubernetes object according to the expectations listed below:

• Docker images registry will be deployed as a single replica;

• Pod runs a single container with image registry :2.7.1;

• Container stores images under /var/iib/registry. Mount a Persistent Volume to store data;

• Additional configuration for the Persistent Volume:

o Set access modes to ReadWriteOnce; o Persistent Volume should request 1Gi storage size;

• The container exposes port 5000;

• The container must have configured a LivenessProbe, which will attempt to open a socket to your container on port 5000 (use tcpsocket);

• Additional configurations on the LivenessProbe:

o Set initial probe delay to 10 seconds;

• The container must have configured a ReadinessProbe, which will attempt to open a socket to your container on port 5000 (use tcpsocket);

• Additional configurations on the ReadinessProbe:

o Set initial probe delay to 10 seconds.

Additionally, write a Service with type LoadBaiancer, which exposes the above deployment on port 5000.

For this test, assume that:

• You will use either statefuiset or Deployment object definition from apiGroup apps/vi to create the Deployment;

• If necessary, you may use Persistentvoiumeciaim from apiGroup vi

• You can use tcpsocket requests in probes;

• The Deployment will be created in the default namespace (it is not expected to define its own namespace);

• Your solution will be applied using kubectl apply -n default -f solution.yaml;

• The file you are editing should be written as a valid YAML file;

The Deployment will run on Kubernetes v1.16.

apiVersion: apps/v1

kind: Deployment

metadata:

name: docker-registry

spec:

replicas: 1

selector:

matchLabels:

app: docker-registry

template:

metadata:

labels:

app: docker-registry

spec:

containers:

- name: registry-container

image: registry:2.7.1

ports:

- containerPort: 5000

volumeMounts:

- name: registry-data

mountPath: /var/iib/registry

volumes:

- name: registry-data

persistentVolumeClaim:

claimName: registry-pvc

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: registry-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

---

apiVersion: v1

kind: Service

metadata:

name: registry-service

spec:

selector:

app: docker-registry

ports:

- protocol: TCP

port: 5000

targetPort: 5000

type: LoadBalancer

kubectl apply -n default -f solution.yaml

SOLUCION BING CHAT

apiVersion: apps/v1

kind: Deployment

metadata:

name: registry-deployment

spec:

replicas: 1

selector:

matchLabels:

app: registry

template:

metadata:

labels:

app: registry

spec:

containers:

- name: registry

image: registry:2.7.1

ports:

- containerPort: 5000

volumeMounts:

- name: registry-storage

mountPath: /var/iib/registry

livenessProbe:

tcpSocket:

port: 5000

initialDelaySeconds: 10

readinessProbe:

tcpSocket:

port: 5000

initialDelaySeconds: 10

volumes:

- name: registry-storage

persistentVolumeClaim:

claimName: registry-pvc

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: registry-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

---

apiVersion: v1

kind: Service

metadata:

name: registry-service

spec:

type: LoadBalancer

ports:

- port: 5000

selector:

app: registry

kubectl apply -n default -f solution.yaml